2012-09

Adaptive Branch and Bound for Efficient Solution of Mixed-Integer Programs Formulated with Big-M

Gan, Eng Kiat Russell

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

ADAPTIVE BRANCH-AND-BOUND FOR EFFICIENT SOLUTION OF MIXED-INTEGER PROGRAMS FORMULATED WITH "BIG-*M*"

by

Eng Kiat Russell Gan

September 2012

Thesis Co-Advisors:	Kevin Wood
	Javier Salmerón
Second Reader:	Nedialko Dimitrov

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 2012	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE Adaptive Branch and Bound for Efficient Solution of Mixed-Integer Programs Formulated with "Big-*M*"		**5. FUNDING NUMBERS**	
6. AUTHOR(S) Eng Kiat Russell Gan			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number _____N/A_____.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		**12b. DISTRIBUTION CODE**	

13. ABSTRACT (maximum 200 words)

This thesis describes three specialized branch-and-bound (B&B) algorithms for solving a mixed-integer program (MIP) that incorporates standard "big-*M*" constructs. The goal is to identify valid values for *M* that also lead to short solution times.

One algorithm initializes large instances of *M* (giving a weak relaxation of the MIP), and decreases these as required to increase efficiency of the standard B&B. Two algorithms initialize small and possibly invalid instances of *M*, and subsequently increase those values in an attempt to ensure solution validity. Each algorithm requires a model-specific test condition to detect weak or invalid *M*'s.

We test all algorithms on an uncapacitated *k*-median problem (a variant of the uncapacitated facility location problem), and one algorithm on a shortest-path interdiction problem (SPIP). We observe substantial reduction in run times in almost all cases tested. When solving for exact solutions, computational results show that the proposed algorithms may reduce solution times by up to 75% for the uncapacitated *k*-median problem and 99% for the SPIP. When the algorithms yield marginally suboptimal solutions, substantial solution-time improvements are also recorded.

While testing is limited, this thesis serves as a proof-of-concept that the proposed adaptive algorithms can be effective in reducing solution times and producing optimal or nearly optimal solutions.

14. SUBJECT TERMS Big-*M*, bound coefficients, uncapacitated facility location problem, shortest-path interdiction problem, branch-and-bound, uncapacitated *k*-median problem.			**15. NUMBER OF PAGES** 67
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

ADAPTIVE BRANCH AND BOUND FOR EFFICIENT SOLUTION OF MIXED-INTEGER PROGRAMS FORMULATED WITH "BIG-*M*"

Eng Kiat Russell Gan
Major, Singapore Army
B.Sc., National University of Singapore, 2005

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September 2012

Author: Eng Kiat Russell Gan

Approved by: Kevin Wood
Thesis Advisor

Javier Salmerón
Thesis Co-Advisor

Nedialko Dimitrov
Second Reader

Robert F. Dell
Chair, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis describes three specialized branch-and-bound (B&B) algorithms for solving a mixed-integer program (MIP) that incorporates standard "big-*M*" constructs. The goal is to identify valid values for *M* that also lead to short solution times.

One algorithm initializes large instances of *M* (giving a weak relaxation of the MIP), and decreases these as required to increase efficiency of the standard B&B. Two algorithms initialize small and possibly invalid instances of *M*, and subsequently increase those values in an attempt to ensure solution validity. Each algorithm requires a model-specific test condition to detect weak or invalid *M*'s.

We test all algorithms on an uncapacitated *k*-median problem (a variant of the uncapacitated facility location problem), and one algorithm on a shortest-path interdiction problem (SPIP). We observe substantial reduction in run times in almost all cases tested. When solving for exact solutions, computational results show that the proposed algorithms may reduce solution times by up to 75% for the uncapacitated *k*-median problem and 99% for the SPIP. When the algorithms yield marginally suboptimal solutions, substantial solution-time improvements are also recorded.

While testing is limited, this thesis serves as a proof-of-concept that the proposed adaptive algorithms can be effective in reducing solution times and producing optimal or nearly optimal solutions.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

The continuous relaxation of a mixed-integer program (MIP) containing standard "big-*M*" constructs can be weak because large values of *M* ("bound coefficients") are required to ensure the model's validity. This, in turn, may lead to slow convergence when solving the model using a standard branch-and-bound (B&B) algorithm. For the purpose of this thesis, we consider bound coefficients to be unsuitable if they are so large solution times may be long, or so small that the model may be invalid. One approach to reduce solution times is to pre-calculate tight and valid bound coefficients prior to solving the MIP. A second approach, which is the focus of this thesis, modifies the standard B&B algorithm to initialize bound coefficients to possibly unsuitable values, but then attempts to recognize and update those coefficients by either increasing or decreasing them as required for validity or short solution times, respectively. This process repeats until the algorithm deems all bound coefficients suitable. The algorithms are termed "adaptive" because they repeatedly update bound coefficients.

Three adaptive algorithms are developed. Two of the algorithms continually update bound coefficients throughout the B&B tree, whereas the other updates bound coefficients only at the root node of the tree, i.e., before any branching occurs. Briefly, the algorithms are:

1. Algorithm 1. This algorithm initializes large (weak) bound coefficients and reduces them throughout the B&B tree as required for efficiency.

2. Algorithm 2. This algorithm initializes small and potentially invalid bound coefficients, but, at the root node only, increases each coefficient that is recognized by the algorithm to be potentially invalid before commencing the exploration of the B&B tree. When updating occurs, the problem is re-solved from the root node with the updated bound coefficients.

3. Algorithm 3. This algorithm initializes small and potentially invalid bound coefficients but, throughout the B&B tree, increases each coefficient that is

recognized by the algorithm to be potentially invalid. When updating occurs, the problem is re-solved from the root node with the updated coefficients.

Central to all algorithms is the need for a model-specific *test condition* that signifies either potentially invalid or weak bound coefficients.

For demonstration purposes, all algorithms are tested on an instance of a weak formulation of the uncapacitated k-median problem. The problem comprises 50 locations, from which 5 must be chosen to fulfill the demands of 100 customers for a single product at minimum cost. Solution times for the proposed algorithms are compared to solution times obtained when the same formulation is solved using the standard B&B with standard, provably valid, bound coefficients. In the instances where Algorithms 1 and 2 yield the optimal solution, we observe a reduction in solution time of up to 75% and 62%, respectively. Algorithm 3 yields the optimal solution for all instances tested, with up to a 37% decrease in solution time. An alternative test condition for Algorithm 3 also yields an optimal solution in all cases, but with excessive solution times.

We test Algorithm 3 on an instance of a shortest-path interdiction problem (SPIP) comprising 50 nodes and 416 arcs. The basic model seeks to maximize the shortest-path length in a network by interdicting arcs and making them impassable to a network user. We solve a standard MIP formulation derived by taking the dual of the inner shortest-path model. Algorithm 3 exhibits up to a 99.8% reduction in solution time. We also observe a maximum 3% deviation from the optimal objective value.

This thesis serves as a proof-of-concept to show that the proposed adaptive algorithms can be used to substantially reduce solution times of MIPs containing standard big-M constructs. Note that, since the solution quality of all proposed algorithms cannot be guaranteed *a priori* for all problem instances, these adaptive algorithms are heuristics. Although results are promising, further research must be conducted before practical implementation can be adopted.

ACKNOWLEDGMENTS

I would like to dedicate this article to my wife, Mandy, and son, Bertrand, who have supported me in every possible way since our arrival at the Naval Postgraduate School. Their sacrifices have allowed me to achieve what I have today.

I am also grateful to Distinguished Professor Kevin Wood and Associate Professor Javier Salmerón for their guidance and patience throughout this journey. I have learnt much from them, and can only hope that I did not disappoint. I would also like to express my gratitude to Assistant Professor Nedialko Dimitrov for his assistance throughout my stay here.

Also, I would like to thank other instructors who have helped me develop over the past year in the areas of optimization, statistics and simulation. These are undoubtedly valuable tools that I will keep with me throughout my career.

Finally, I would like to thank the Singapore Armed Forces for providing me with this opportunity to further my studies at the Naval Postgraduate School.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

This thesis describes specialized branch-and-bound (B&B) algorithms for solving a mixed-integer program (MIP) in which binary variables x_i control continuous variables y_i representing "activity levels": For each *i,* either both variables have values 0, or the controlling binary variable becomes 1 and the continuous activity is then allowed to take on positive values, up to the value of its bound coefficient. In particular, we are interested in MIPs that contain constraints of the form:

$$y_i - M_i x_i \leq 0, \quad \forall i \in I \tag{1.1}$$

$$y_i \geq 0, \quad \forall i \in I \tag{1.2}$$

$$x_i \in \{0,1\}, \quad \forall i \in I, \tag{1.3}$$

where $M_i > 0$, for all $i \in I$. The model coefficient M_i is the bound coefficient for the continuous activity level y_i, and x_i is the controlling binary variable. Note that y_i can actually correspond to an affine function of other continuous variables. The key characteristic of the choice of M_i is that this bound coefficient must be chosen such that (1.1) is redundant when x_i is 1.

Examples of models that may contain "big-*M*" constructs are weak formulations of the uncapacitated facility location problem (UFLP), as described by Khumawala (1972), and the closely related uncapacitated *k*-median problem (Lin & Vitter, 1992). Similar constructs appear in production-scheduling problems, e.g., Eppen and Martin (1987), Maes and Van Wassenhove (1988), and MIP formulations of min-max (or max-min) models such as network-interdiction problems, e.g., Brown et al. (2006), Wood (1993). In all problems, "variable upper bounding constraints," i.e., equation (1.1), connect binary and continuous variables.

The continuous relaxation of a "big-*M* model" can be weak because large values of *M* are required to ensure the model's validity. Some textbooks such as Markland and Sweigart (1987, p. 505) advocate using "a large number" for bound coefficients in the

model. However, Camm et al. (1990) note that, by making bound coefficients large, the feasible region of the continuous relaxation is expanded unnecessarily. This, in turn, may lead to slow convergence when solving the model using a standard B&B algorithm. Other textbooks, such as Rardin (1998, p. 642), require bound coefficients to be "sufficiently large," i.e., not excessively large for efficiency, but large enough for validity. Rardin emphasizes that "whenever a discrete model requires sufficiently large big *M*'s, the strongest relaxations will result from models employing the smallest valid choice of those constants." For the purpose of this thesis, bound coefficients are considered *unsuitable* if they are so small that the model may not be valid, or so large that solution times are excessive.

One approach to reduce solution times for a "big-*M* model" is to predefine valid values for the bound coefficients prior to solution, while attempting to make those values as tight as possible. Crowder et al. (1983), Van Roy and Wolsey (1987) and Johnson et al. (1985) provide preprocessing techniques to develop tighter values for bound coefficients. These early techniques primarily involve careful model formulation, and are widely considered as standard present-day modeling techniques. For example, in an UFLP, using the sum of demands for bound coefficients is more efficient than using an arbitrarily large value. These tighter formulations, however, may still be inefficient for solving large problems. We note that there may be stronger preprocessing techniques available that are unknown to this author.

This thesis explores a second approach to reduce solution times for a "big-*M* model." The goal is to modify the standard B&B algorithm to initialize bound coefficients to possibly unsuitable values, but then to recognize and update those coefficients by either increasing or decreasing them as required for validity or efficient solution, respectively. We propose three adaptive algorithms; two algorithms continually update bound coefficients throughout the B&B tree, whereas the other updates bound coefficients only at the root node of the tree, i.e., before any branching occurs. The algorithms are termed "adaptive" because they repeatedly update bound coefficients. The solution quality of the algorithms, at the time of writing, cannot be guaranteed *a priori*. Hence, these adaptive algorithms are heuristics.

A. STANDARD BRANCH-AND-BOUND

B&B implicitly searches the entire feasible region for the continuous relaxation of the MIP it is solving. Every branch partitions the locally restricted feasible region into three parts, one of which cannot contain an integer solution and is therefore not explored. Assuming a minimization problem, every existing partitioned region with an objective value that exceeds the objective value of any known integer solution cannot contain the optimal solution; therefore it is terminated or *fathomed*. Similarly, partitioned regions that are infeasible are also fathomed. A lower bound is also derived from the smallest objective value of continuous solutions from the existing partitioned regions. The partitioning continues until an integer solution is found such that its objective value is equal to the lower bound. See Lawler and Wood (1966) for a survey of the B&B algorithm and its applications.

B. LITERATURE REVIEW

The adaptive algorithms proposed in this thesis, while not exactly analogous, are similar in flavor to variable-bound tightening techniques used in multi-start methods for solving non-linear programs (NLPs); see Chinneck (2002). Multi-start methods for an NLP aim to provide an optimization algorithm with an initial solution from which it can reach feasibility efficiently. These methods furnish random starting solutions within a region provided by the variable-bounds. However, the variable-bounds provided by the original model may define a region that is much larger than required given the other constraints in the problem. Therefore, it is desirable to define a region that is not excessively larger than the region of interest. Tightening of variable-bounds is then applied to give the multi-start algorithms a smaller region from which a good initial solution is likely to be found efficiently. Chinneck notes that the probability of obtaining the tightest possible variable-bounds that do not eliminate feasible solutions is small. He proposes bound-tightening techniques via the following means:

1. Depending on the modeler's aims and analysis of the problem, the bounds may be adjusted manually to contain suspected optimal points or simply to exclude regions that are unlikely to contain an optimal solution.

2. Presolving techniques, such as those proposed by Brearley et al. (1975), can be used to tighten the variable-bounds or even fix the value of certain variables.

3. Linear and non-linear interval analysis can be applied when the bounds can be inferred from constraints present in the model. The idea is to determine the maximum possible range for each variable v, by considering the constraints where v appears and the bounds on other variables present in those constraints. This is also a useful method to detect infeasibility of a model should the maximum possible range for any variable fall outside the stated range in the original problem.

Chinneck also proposes an alternative method to bound variables in an NLP. This method starts with a small region and expands it in several stages. However, it is possible that the final bounds derived may be invalid, resulting in suboptimal solutions.

Chinneck describes how variable-bounds can be increased and decreased to obtain a good initial point efficiently. In a similar fashion, an adaptive B&B algorithm will increase or decrease bound coefficients to help improve efficiency; rules for changing coefficients will attempt to ensure validity.

C. ADAPTIVE BRANCH AND BOUND

The adaptive B&B algorithms described in this thesis explore these three distinct methods:

1. Initialize each bound coefficient to a large and probably weak value pre-calculated via standard techniques. Such coefficients are henceforth termed *baseline coefficients*. Then, during the exploration of the B&B tree, reduce each coefficient that is recognized by the algorithm to be potentially weak.

2. Initialize each bound coefficient to a small value that is potentially invalid. Then, at the root node only, increase each coefficient that is recognized by the algorithm to be potentially invalid before commencing the exploration of the B&B tree.

3. Initialize each bound coefficient to a small value that is potentially invalid. Then, throughout the B&B search, increase each coefficient that is recognized by the algorithm to be potentially invalid. Whenever updating occurs, the problem is re-solved *from the root node* with the updated bound coefficients.

The algorithms recognize a potentially unsuitable bound coefficient at a node when a *test condition* is met, and attempt to increase or decrease the coefficient depending on the method being implemented. Test conditions are model-specific, and when met, signify that a coefficient is potentially invalid or weak. A test condition that identifies a potentially invalid bound coefficient is referred to as an "invalidity test condition," and a test condition that identifies a potentially weak bound coefficient is referred to as an "inefficiency test condition." Once the potentially unsuitable bound coefficients are updated, the adaptive algorithm continues until a set of bound coefficients is deemed unsuitable at a subsequent step in the algorithm.

The adaptive B&B algorithms are described in detail in Chapter II, and are applied to a weak formulation of the uncapacitated k-median problem in Chapter III and a shortest-path interdiction problem (SPIP) in Chapter IV. (The aforementioned weak formulation would probably not be used in practice, and is used in this thesis only because it provides a simple, paradigmatic test problem.) This thesis conducts a proof-of-concept study to show that the proposed adaptive algorithms can be used to reduce solution times of MIPs containing constraints of the form (1.1) – (1.3). Positive results on the test problems will indicate that further research in this area is warranted.

THIS PAGE INTENTIONALLY LEFT BLANK

II. ADAPTIVE BRANCH-AND-BOUND ALGORITHMS

This chapter describes the three variants of adaptive B&B algorithms in further detail.

The following notation is common to all three algorithms:

n node in the B&B tree; $n = 0$ is the root node

M_i^I initial value of bound coefficient used in the *i*-th constraint

M_i adjusted value of bound coefficient used in the *i*-th constraint

I_n^v index set of "violated" constraints, i.e., constraints whose coefficients M_i meet the inefficiency or invalidity test condition, at node *n*.

Note that the details of the inefficiency and invalidity test conditions are not specified until the test models are introduced in Chapters III and IV.

A. ALGORITHM 1: REDUCING A LARGE, INITIAL *M*

Algorithm 1 assumes that large, valid baseline bound coefficients have been pre-calculated via standard techniques. The algorithm seeks to recognize weak coefficients and decrease their values appropriately. The process of updating bound coefficients can occur anywhere in the B&B tree.

We define parameter β, $0 < \beta < 1$, as the *coefficient reduction factor*, where $1 - \beta$ is the factor by which a bound coefficient is reduced each time an inefficiency test condition is met. Algorithm 1 follows:

Algorithm 1	
Input:	A MIP containing constraints of the form (1.1), (1.2) and (1.3). Coefficient reduction factor β, $0 < \beta < 1$. Inefficiency test condition with associated parameters. (See equation 3.B.1) Baseline coefficients M_i^I, $\forall i \in I$.
Output:	A feasible solution to the MIP.
Step 0	Start a standard B&B algorithm; Solve root node $n = 0$;
Step 1	Select an active node *n* to process using the B&B algorithm's selection procedure; If node *n* is fathomed then go to step 4;
Step 2	Determine I_n^v via inefficiency test condition; If $I_n^v = \varnothing$ then let the B&B algorithm continue, possibly creating child nodes, and then proceed to step 1; If $I_n^v \neq \varnothing$ then go to step 3;
Step 3	Let $M_i \leftarrow \beta^{depth(n)} M_i^I$, $\forall i \in I_n^v$, where $depth(n)$ is the depth of the node *n* being solved within the B&B tree; Impose M_i, $\forall i \in I$ on any descendants of node *n*;
Step 4	If there exists an active node in the B&B tree then go to step 1;
Step 5	Print solution; Stop;
End of Algorithm 1	

Remark: By observation, a larger value for β implies that M_i is reduced at a slower rate than if a smaller β is used.

Ideally, the incumbent value for the bound coefficient would be used in the updating process, i.e., $M_i \leftarrow \beta M_i$. However, this value could not be retrieved during the execution of the B&B algorithm using Xpress software (see Section D). Hence, scaling according to depth in the B&B tree serves as a proxy.

The reader should note that the algorithm is a heuristic because any bound coefficient may be reduced below its smallest valid value.

B. ALGORITHM 2: INCREASING A SMALL, INITIAL *M* AT ROOT NODE

The second algorithm attempts to identify invalid bound coefficients based on the root-node solution, that is, before any branching occurs. Using a small and possibly invalid initial value for each bound coefficient, the relaxed problem is solved, and if the algorithm detects potentially invalid coefficients, these coefficients are increased and the root node is re-solved. This process continues until the algorithm does not detect any potentially invalid bound coefficients. When this occurs, the standard B&B algorithm proceeds normally (without further updates of the coefficients).

We define $\gamma > 1$ to be the *increase factor* of the bound coefficient, and update $M_i \leftarrow \gamma M_i$ for all $i \in I_0^v$. Algorithm 2 follows:

Algorithm 2	
Input:	A MIP containing constraints of the form (1.1), (1.2) and (1.3). Increase factor $\gamma > 1$ (for processing the root node $n = 0$). Invalidity test condition with associated parameters (for processing the root node $n = 0$). (See equation 3.B.2) Small value of $M_i^I,\ \forall i \in I$.
Output:	A feasible solution to the MIP defined with weak bound coefficients.
Step 0	$M_i \leftarrow M_i^I,\ \forall i \in I$;
Step 1	Solve root node $n = 0$;
Step 2	Determine I_0^v via invalidity test condition; If $I_0^v = \varnothing$ then go to step 4; If $I_0^v \neq \varnothing$ then go to step 3;
Step 3	Set $M_i \leftarrow \gamma M_i,\ \forall i \in I_0^v$; Go to step 1;
Step 4	Implement a standard B&B algorithm;
Step 5	Print solution; Stop;
End of Algorithm 2	

This algorithm is optimistic as it relies only on root node solutions in the attempt to derive valid bound coefficients.

C. ALGORITHM 3: INCREASING A SMALL, INITIAL *M* WITHIN THE SEARCH TREE

In Algorithm 2, bound coefficients are increased at the root node only. The last algorithm, Algorithm 3, allows bound coefficients to be increased anywhere in the B&B

search tree, including the root node. To do this, Algorithm 3 leverages on Algorithm 2 to process the root node $n = 0$; other inputs are defined as required to process tree nodes $n \geq 1$. Should Algorithm 3 detect that incumbent bound coefficients are possibly invalid, these coefficients are increased and the problem is re-solved *from the root node*. This process continues until the standard B&B algorithm converges with all bound coefficients deemed valid.

Define $\varphi > 1$ to be the increase factor in the B&B tree ($n \geq 1$), similar to how γ is defined in Algorithm 2. Algorithm 3 follows:

Algorithm 3	
Input:	Same inputs as for Algorithm 2 (for processing the root node $n = 0$). Increase factor $\varphi > 1$ (for processing tree nodes $n \geq 1$). Invalidity test condition with associated parameters (for processing tree nodes $n \geq 1$.) (See equations 3.B.3, 3.B.4 and 4.B.1)
Output:	A feasible solution to the MIP defined with weak bound coefficients.
Step 0	Implement Algorithm 2 until $I_0^v = \varnothing$, i.e., until standard B&B proceeds normally;
Step 1	Select an active node n to process using the B&B algorithm's selection procedure; Determine I_n^v via invalidity test condition; If $I_n^v = \varnothing$ then go to step 3; If $I_n^v \neq \varnothing$ then go to step 2;
Step 2	Set $M_i \leftarrow \varphi M_i$, $\forall i \in I_n^v$; Go to step 0;
Step 3	If there exists an active node in the B&B tree then go to step 1;
Step 4	Print solution; Stop;
End of Algorithm 3	

As opposed to Algorithm 2, which operates only at the root node, this algorithm provides the flexibility to adjust bound coefficients as the search tree grows. This increases the likelihood of attaining an optimal solution as the bound coefficient values required for validity might not be obtained by iterating at the root node alone.

D. IMPLEMENTATION OF ALGORITHMS

All models and algorithms are implemented using Xpress-IVE Version 1.22.04 as the solving engine. Computations are performed on an Acer Aspire 4741G laptop with 6 gigabytes of RAM, running on an Intel® Core™ i5-M460 processor (2.53GHz, 3MB L3 cache) and Windows® 7 operating system.

For the purpose of evaluating the performance of the adaptive algorithms against a standard B&B algorithm, we disable all of Xpress-IVE's presolve techniques, cutting-planes strategies, and heuristics.

THIS PAGE INTENTIONALLY LEFT BLANK

III. TEST MODEL 1: UNCAPACITATED *k*-MEDIAN PROBLEM

This chapter describes the weak formulation of the uncapacitated *k*-median problem and solves it using Algorithms 1, 2 and 3. We state the test conditions for and analyze the results of each algorithm.

A. MODEL FORMULATION

In the uncapacitated *k*-median problem, facilities of unrestricted size are placed among *n* possible locations with the objective of minimizing cost for satisfying given demands for a single product at *m* customer locations. Costs include a per-unit shipping cost for product shipped from facility *i* to customer *j*, and a per-unit penalty at each customer location for any unmet demand. The number of facilities to be opened is given.

We note that the uncapacitated *k*-median problem is a variant of the UFLP. The difference is that UFLP applies a fixed cost for opening any facility, rather than limiting the number of opened facilities. See Reese (2005) for a summary of solution methods for the uncapacitated *k*-median problem.

As highlighted in Chapter I, the uncapacitated *k*-median problem formulation presented here is a weak version that is used for testing purposes; this formulation might not be used in practice. (Efroymson & Ray 1966 do study the weak formulation, however.) The differences between the weak and strong formulations are discussed at the end of this section.

The mathematical-programming formulation of the uncapacitated *k*-median problem follows:

Indices and Index Sets

$i \in I$ facilities ($I = \{i_1, ..., i_n\}$)

$j \in J$ customers ($J = \{j_1, ..., j_m\}$)

Parameters

d_j units of demand at customer *j* for some commodity

M_i bound coefficient for total shipping from facility *i*

c_{ij} per-unit shipping cost from facility *i* to customer *j*

p_j per-unit penalty of unmet demand at customer *j*

k exact number of facilities that must be opened

Remark: At convergence of the adaptive algorithms, the bound coefficients M_i should be large enough such that no unmet demand arises. However, we acknowledge that some of the bound coefficients may be invalid, at least temporarily, as the adaptive B&B algorithms proceed.

Variables

x_i 1 if facility *i* is opened, 0 otherwise

y_{ij} number of units shipped from facility *i* to customer *j*

s_j units of unmet demand at customer *j*

Formulation

$$z^* = \min_{\mathbf{x},\mathbf{y},\mathbf{s}} \sum_i \sum_j c_{ij} y_{ij} + \sum_j p_j s_j \qquad (3.A.1)$$

$$\text{s.t.} \quad \sum_j y_{ij} \leq M_i x_i, \quad \forall i \in I \qquad (3.A.2)$$

$$\sum_i y_{ij} + s_j \geq d_j, \quad \forall j \in J \qquad (3.A.3)$$

$$\sum_i x_i = k \qquad (3.A.4)$$

$$\mathbf{y} \geq \mathbf{0} \qquad (3.A.5)$$

$$\mathbf{s} \quad \geq \quad \mathbf{0} \tag{3.A.6}$$

$$\mathbf{x} \quad \in \quad \{0,1\}^{n \times m} \tag{3.A.7}$$

The objective function (3.A.1) defines the total cost to be minimized. This comprises transportation costs and penalties should any unfulfilled demand occur. Constraints (3.A.2) govern the relationship between production and setup of a facility.

Constraints (3.A.3) determine the amount of unfulfilled demand and charge the corresponding penalty to the objective function (3.A.1). Note that, while fixing the number of facilities to be opened in (3.A.4) makes this problem an uncapacitated *k*-median problem, we believe the proposed algorithms can be applied to weak UFLP formulations, also.

Remark: The strong formulation presented by Balinski (1966) replaces constraints in (3.A.2) with $y_{ij}/d_j \leq x_i$ for all $i \in I$, and $j \in J$, where y_{ij}/d_j represents the fraction of customer *j*'s demand that is satisfied by facility *i*..

B. TEST CONDITIONS FOR UNCAPACITATED *k*-MEDIAN PROBLEM

This section discusses possible test conditions, specific for the uncapacitated *k*-median problem, to detect potentially weak or invalid bounds. We first present data for a small problem that is used to explain the implementation of the algorithms. Actual computational tests are carried out on a larger problem (see Section C).

1. Illustrative Example Data

Consider four possible facility locations and five customers. Two facilities must be opened. Each of the five customers' demands is shown in Table 1.

Customer	**1**	**2**	**3**	**4**	**5**
Demand, d_j	180	240	210	300	150

Table 1. Customer demands

Table 2 presents per-unit transportation costs c_{ij} for shipments from facility *i* to customer *j*.

Facility \ Customer	1	2	3	4	5
1	2	5	3	2	2
2	1	3	5	4	2
3	5	3	1	4	3
4	2	5	2	1	3

Table 2. Per-unit transportation costs

For each customer, penalty costs per unit of unmet demand are set to be higher than the maximum of all transportation costs. Specifically, we set $p_j = 6$ for all *j*.

For this problem, an optimal solution is **x*** = (0, 1, 0, 1) with an optimal objective value *z** = 1920.

2. Algorithms with Test Conditions

a. Test Condition for Algorithm 1

Starting with the baseline coefficient, $M_i \leftarrow M_i^I$, we infer that a bound coefficient is possibly weak whenever the total demand satisfied by facility *i* is less than a predefined proportion of M_i. Let $depth(n)$ denote the depth of node *n* in the B&B tree being processed. The inefficiency test condition for Algorithm 1 applied to the uncapacitated *k*-median problem is

$$i \in I_n^v \text{ if } \sum_j y_{ij} \le \alpha\beta^{depth(n)-1} M_i^I, \tag{3.B.1}$$

where $\alpha < 1$ is the *inefficiency test factor* that defines the aforementioned proportion. After applying the test condition to compute I_n^v, we let $M_i \leftarrow \beta^{depth(n)} M_i^I$, for all $i \in I_n^v$, and Algorithm 1 continues.

We note that, if the incumbent value for the bound coefficient could be retrieved in Xpress-IVE, the test condition above would be replaced by $\sum_j y_{ij} \le \alpha M_i$, where M_i is the relevant bound coefficient. We also note that, in order for the algorithm to reduce a bound coefficient M_i, the total flow from facility *i* cannot exceed $M_i \alpha$. Hence, a small inefficiency test factor α implies a smaller I_n^v and is a restriction compared to a larger value of α. The larger β is, the slower the coefficient is reduced.

To illustrate, we now solve the uncapacitated *k*-median example using Algorithm 1 with the following inputs:

Input: Uncapacitated *k*-median problem defined in previous section.

Coefficient reduction factor $\beta = 0.9$.

$M_i^I = \sum_j d_j = 1080, \quad \forall i \in I$

Inefficiency test condition $\sum_j y_{ij} \le \alpha \beta^{depth(n)-1} M_i^I$, with $\alpha = 0.2$

Step 0: A standard B&B algorithm solves the root node. The solution is $\mathbf{x}^T = (1.00,\ 0.528,\ 0.194,\ 0.278)$, and the amount each facility supplies to customers is shown in Table 3.

Step 1: Since only the root node has been solved and the solution is continuous, it is not fathomed. Algorithm 1 proceeds to process the root node.

Step 2: Facilities $i = 1$ and $i = 3$ meet the inefficiency test condition (see Table 3), thus $I_o^v = \{1, 3\}$.

Step 3: Let $M_i \leftarrow \beta M_i^I = 0.9 \times 1080 = 972$, $\forall i \in I_0^v$, and impose the updated values on all descendant nodes in the B&B tree.

Step 4: The B&B algorithm's selection procedure selects one of the two active nodes (i.e., children nodes of the root node) to process, and the adaptive algorithm continues.

Continuing this process, an optimal solution obtained is **x*** = (0, 1, 0, 1), z^*=1920.

Customer / Facility	1	2	3	4	5	Total	$\sum_j y_{ij} \le 216$?
1					150	150	Yes
2	180	240				420	No
3			210			210	Yes
4				300		300	No

Table 3. Root node solution for flow variables (Algorithm 1)

b. Test Condition for Algorithm 2

When starting with a small bound coefficient, Algorithm 2 detects potential invalidity whenever the solution at the root node shows that the total demand satisfied by facility i is at least a predefined proportion δ of the incumbent bound coefficient. The invalidity test condition for Algorithm 2 applied to the uncapacitated *k*-median problem is

$$i \in I_0^v \text{ if } \sum_j y_{ij} \ge \delta M_i , \tag{3.B.2}$$

where $\delta \le 1$ is the *invalidity test factor*. After applying the test condition to compute I_0^v, we let $M_i \leftarrow \gamma M_i$, for all $i \in I_0^v$, and the problem is restarted. Furthermore, we set $M_i \leftarrow \sum_j d_j$, for all $i \in I_0^v$ such that $\gamma M_i \ge \sum_j d_j$. (In a similar argument used for the test factor α in Algorithm 1, a larger δ here implies a more stringent condition to signify possible invalidity of M_i.)

To illustrate, we now solve the uncapacitated *k*-median example using Algorithm 2 with the following inputs:

Input: Uncapacitated *k*-median problem defined in previous section.

Increase factor $\gamma = 0.1$ applied at root node.

Invalidity test condition $\sum_j y_{ij} \geq \delta M_i$, with $\delta = 0.3$.

$M_i^I = \sum_j d_j \big/ k = 540, \ \ \forall i \in I$.

The rationale for using the average demand fulfilled by an opened facility as the initial value for bound coefficients is discussed in a subsequent section.

Step 0: Set $M_i \leftarrow M_i^I$, for all $i \in I$.

Step 1: The solution to the root node is $\mathbf{x}^T = (0.278, \ 0.778, 0.389, 0.556)$ and the amount each facility supplies to the customers is shown in Table 4.

Step 2: Facilities i = 2, 3, 4 meet the invalidity test condition (see Table 4), and thus $I_0^v = \{2,3,4\}$.

Step 3: Let $M_i \leftarrow \gamma M_i = 1.1 \times 540 = 594, \ \forall i \in I_0^v$.

The root node is solved again using the updated values for M_2, M_3 and M_4, and checked using the invalidity test conditions. Continuing this process, an optimal solution obtained is **x*** = (0, 1, 0, 1), *z**=1920.

Customer / **Facility**	**1**	**2**	**3**	**4**	**5**	**Total**	$\sum_j y_{ij} \geq 162$?
1					150	150	No
2	180	240				420	Yes
3			210			210	Yes
4				300		300	Yes

Table 4. Root node flow variables solution to MIP defined with invalid bound coefficients (Algorithm 2)

c. Test Condition for Algorithm 3

The invalidity test condition for Algorithm 3, applied to the uncapacitated *k*-media problem, is similar to that of Algorithm 2:

$$i \in I_n^v \text{ if } \sum_j y_{ij} \geq \phi M_i \,, \tag{3.B.3}$$

where $\phi \leq 1$ is the invalidity test factor (similar to δ in Algorithm 2's invalidity test condition). As we have described how Algorithm 2 processes node $n = 0$, we proceed to describe the processing of nodes $n \geq 1$. After applying the test condition to compute I_n^v, we let $M_i \leftarrow \varphi M_i$ for all $i \in I_n^v$ and the problem is restarted at $n = 0$. Furthermore, we set $M_i \leftarrow \sum_j d_j$, for all $i \in I_n^v$, such that $M_i \geq \sum_j d_j$.

Using $\gamma = \varphi = 1.2$, $\delta = 0.1$ and $\phi = 0.9$, the example is solved using Algorithm 3. An optimal solution obtained is **x*** = (0, 1, 0, 1), *z**=1920.

d. Alternative Test Condition for Algorithm 3

We now present an alternative method for detecting potential invalidity in Algorithm 3. We showed earlier how bound coefficients in this algorithm are deemed potentially invalid when the total demand satisfied by facility *i* is at least a predefined proportion of the bound coefficient. A second method of detecting possible infeasibility of bound coefficients lies in the unique structure of an uncapacitated *k*-median problem: Assuming no two transportation costs to a given customer are identical, i.e., $c_{ij} \neq c_{i'j}$, for all $i,\ i',\ i \in I,\ i \neq i'$, for all $j \in J$, then at any node, if the total number of positive flows from facilities to customers is not equal to the number of customers, then the bound coefficients are invalid. This condition implies that that, for any instance of the problem without two identical costs, an optimal solution to every node requires each customer to be serviced by exactly one facility.

Lemma 1: At each node, let *S* be the subset of J such that $j \in S$ if and only if $y_{ij}^* > 0$, where $\mathbf{y}^*$ is the vector of optimal flow from facilities to customers. If $c_{ij} \neq c_{i'j}$, for any $i \neq i' \in I$, and any $j \in J$, then $|S| = |J|$.

Proof: By contradiction, suppose $|S| > |J|$. Then there exists at least one customer whose demand is satisfied by at least two facilities, i.e., $\exists\ \tilde{j}$ such that $\left|\{y_{i\tilde{j}}^* \mid y_{i\tilde{j}}^* > 0\}\right| > 2$. Because $c_{ij} \neq c_{i'j}$ for any $i \neq i' \in I$, and any $j \in J$, we can reduce the objective value by increasing the bound coefficient of facility $\hat{i}$ where $\hat{i} = \underset{i}{\operatorname{argmin}}\{c_{i\tilde{j}} \mid y_{i\tilde{j}}^* > 0\}$, and assigning all demand $d_{\tilde{j}}$ to facility $\hat{i}$. Therefore, $\mathbf{y}^*$ is not optimal.

Now, suppose $|S| < |J|$. This implies that the total capacity of the opened facilities is less than the total demand, and hence unmet demand occurs for some customers. Since $p_j > c_{ij}$ for all $i \in I$, $j \in J$, the objective value can be reduced by increasing bound coefficients. Therefore, $\mathbf{y}^*$ is not optimal.

Hence, at optimality, $|S| = |J|$ holds. **Q.E.D.**

Remark 1: If there exists $i \neq i'$ with $c_{i\tilde{j}} = c_{i'\tilde{j}}$ for some $\tilde{j} \in J$, and $\{i, i'\} \subset \underset{i''}{\operatorname{argmin}}\{c_{i''\tilde{j}} \mid y_{i''\tilde{j}}^* > 0\}$, then we can let the demand at customer $\tilde{j}$ be satisfied by either facility *i* or *i'*. In this case, $|S| = |J|$ is not guaranteed.

Remark 2: Lemma 1 is only a necessary condition for optimality. For sufficiency, we also need to ensure, at nodes where $|S| = |J|$, that $s_j = 0$, for all $j \in J$, i.e., there is no unmet demand. Consider the simple counter-example depicted in Figure 1, where $k = 2$. Suppose the incumbent values for bound coefficients are as shown, and that $c_{ij} = 1$ for all $i \in I$, $j \in J$.

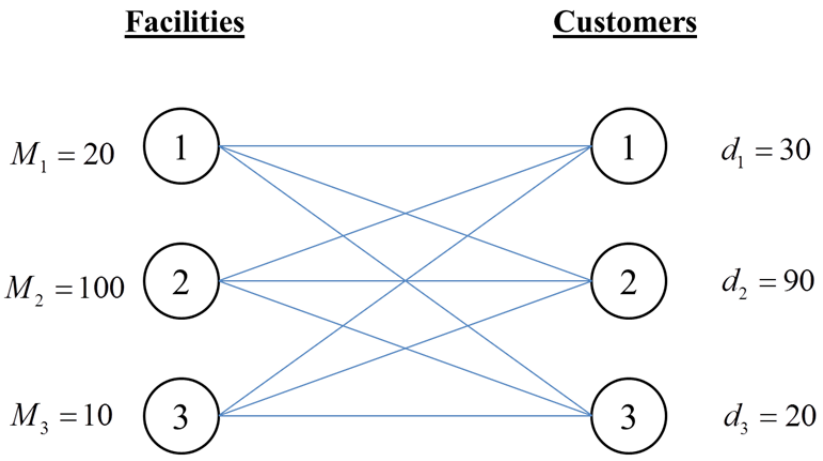

Figure 1. Bipartite graph for illustrating Lemma 1.
This graph depicts an instance where an optimal solution of $\mathbf{x}^* = (1,\ 1,\ 0)$ with $y_{13} = 20$, $y_{21} = 30$ and $y_{22} = 70$ does not meet the invalidity test condition of $|S| \neq |J|$, but still yields a suboptimal solution to the original MIP.

An optimal solution to this problem is $\mathbf{x}^* = (1,\ 1,\ 0)$, with $y_{13} = 20$, $y_{21} = 30$ and $y_{22} = 70$. Observe that $|S| = |J|$. However, this solution is clearly suboptimal since there is unmet demand occurring for customer 2. Hence, a sufficient condition would also demand that $s_j = 0$, for all $j \in J$, whenever $|S| = |J|$.

One approach to ensure optimality using only the sufficient condition is to initialize bound coefficients to the average demand fulfilled by an opened facility, i.e., $M_i \leftarrow M_i^I = \sum_j d_j / k$, for all $i \in I$. In that manner, unmet demand will never occur.

We utilize Lemma 1 as another invalidity test; that is, at any node, whenever $|S| \neq |J|$, we adjust the bound coefficients for all facilities that are servicing a customer. The alternative invalidity test condition for Algorithm 3 applied to the uncapacitated *k*-median problem is

$$i \in I_n^v \text{ if } \sum_j y_{ij} \geq 0 \text{ and } |S| \neq |J| \qquad (3.B.4)$$

We solve the illustrative example using this alternative test condition, and the following inputs:

Input: Uncapacitated *k*-median problem defined in previous section.

Growth factor $\gamma = \varphi = 2$.

Invalidity test condition $|S| \neq |J|$.

$M_i^I = \sum_j d_j / k = 540, \quad \forall i \in I$.

Solving, Algorithm 3 with the alternative test condition yields the optimal solution **x*** = (0, 1, 0, 1), *z**=1920.

C. COMPUTATIONAL RESULTS

We test the algorithms on one randomly generated instance of the uncapacitated *k*-median problem with $n = 50$ locations, from which $k = 5$ must be chosen to fulfill the demands of $m = 100$ customers. Data for the model are generated using the uniform probability distributions specified in Appendix A.

The test problem is first solved using a standard B&B algorithm, and the optimal solution and time taken to solve are recorded. We then use the proposed algorithms to solve the test problem. The measures of effectiveness are:

1. The relative optimality gap, possibly 0%, between the optimal solution derived from the standard B&B algorithm with baseline (weak) coefficients, and the solution derived from the proposed algorithms; and

2. The time taken by the proposed algorithms to complete, as compared to the time taken by the standard B&B algorithm with baseline coefficients.

1. Solution by Standard Branch and Bound Algorithm

The test problem is first run using the standard B&B algorithm, once with baseline (weak) coefficients, and once with tighter, empirically valid, bound coefficients. Henceforth, the test model with weak bound coefficients is called the "base-case." Table 5 shows the results.

Case	Bound Coefficient, M_i	Solution Time (s)	Optimality Gap (%)
Base-case: Weak Bound Coefficient	$\sum_j d_j$	1,136	0.0
Tight Bound Coefficient	$0.3\sum_j d_j$	116	0.0

Table 5. Solution times for standard B&B on the uncapacitated *k*-median problem

As discussed in Chapter I, having weak bound coefficients can increase solution times. In this instance, the time taken to solve the problem using tight bound coefficients is about 10% of the time taken to solve the problem using baseline coefficients. Naturally, tight bound coefficients are, in general, unknown.

2. Algorithm 1: Results and Analysis

For select values of α and β, we proceed to run Algorithm 1 with $M_i^I \leftarrow \sum_j d_j$ Table 6 displays the results.

Test Factor α	Decrease Factor β	Solution Time (s)	Number of Bound Coefficient Updates	Optimality Gap (%)
0.1	0.1	2.1	22,624	312.2
	0.2	2.3	23,961	5.1
	0.3	1.7	16,771	2.5
	0.4	6.2	60,199	2.7
	0.5	10.0	99,719	2.4
	0.6	13.3	97,142	0.57
	0.7	36.2	188,624	1.2
	0.8	94.9	526,975	0.5
	0.9	288.7	1,071,470	0.0
0.2	0.1	0.6	1,551	315.2
	0.2	0.3	714	57.3
	0.3	0.3	1,153	127.1
	0.4	0.4	1,363	21.8
	0.5	0.4	1,636	30.1
	0.6	0.5	2,642	31.6
	0.7	0.9	6,162	11.3
	0.8	3.4	27,312	4.7
	0.9	64.6	316,758	1.6
Base-case solution time: 1,136 seconds				

Table 6. Solution times for Algorithm 1 on the uncapacitated *k*-median problem

The algorithm yields the correct solution only when the appropriate parameters are chosen, in this instance, $\alpha = 0.1$ and $\beta = 0.9$. The solution time for this setting of parameters reduces by 75% over the base-case. When $\alpha = 0.2$, we notice that the optimal solution is never obtained, regardless of the decrease factor. Since any inefficiency test factor greater than 0.2 results in weaker testing criteria, no runs beyond $\alpha = 0.2$ are performed.

For tests that yield incorrect solutions, the optimality gap ranges from 0.5% to 312.2%. In certain cases, such as when $\alpha = 0.1$ and $\beta = 0.3$, the algorithm takes 1.7 seconds, and yields a solution with an optimality gap of 2.5%. That means a reduction in computational time of 2.8 orders of magnitude with respect to the base-case, with only modest solution error.

We notice from Table 6 that smaller values of α and larger values of β tend to yield correct, or close-to-correct, solutions. With a small α, when the inefficiency condition is met for a given facility, the total outflow from that facility is small compared to the value of the controlling bound coefficient. This indicates that the incumbent bound coefficient may be too large. And, with a large decrease factor β, given that potential inefficiency is detected, the bound coefficient is not reduced as much as would be using a small β. This limits the chance of eliminating the optimal solution from the solution space. Therefore, using a conservative (small) test factor α and a large decrease factor β increases the probability of Algorithm 1 yielding the correct solution.

3. Algorithm 2: Results and Analysis

Here we test Algorithm 2 for select values of parameters δ and γ, with all M_i^I set to the average demand fulfilled by an opened facility, i.e., $M_i \leftarrow M_i^I = \sum_j d_j \big/ k$, for all $i \in I$. Table 7 presents results.

Test Factor δ	Growth Factor γ	Solution Time (s)	Number of Bound Coefficient Updates	Optimality Gap (%)
0.03	1.1	973.5	17	0.0
	1.5	800.4	4	0.0
0.04	1.1	434.1	4	0.0
	1.5	518.8	17	0.0
0.05	1.1	291.6	17	0.3
	1.5	434.6	17	0.3
0.10	1.1	33.3	17	0.3
	1.5	35.1	4	0.3
0.20	1.1	19.1	11	2.1
	1.5	21.1	4	1.9
Base-case solution time: 1,136 seconds				

Table 7. Solution times for Algorithm 2 on the uncapacitated *k*-median problem

Algorithm 2 yields an incorrect solution for $\delta > 0.04$. In the instances when the appropriate factors are chosen and the algorithm yields the correct solution, solution times are reduced by up to 62%.

For some choices of parameters that do not yield an optimal solution, we observe that the solution times are reduced by as much as 98%, but with an optimality gap of only about 2%.

One reason for the considerably shorter solution times is that Algorithm 2 does not restart the B&B search process once the standard B&B algorithm progresses beyond the root node. Once Algorithm 2 determines that invalidity test conditions are no longer met at the root node, the bound coefficients are fixed. However, it is for precisely this reason that Algorithm 2 may sometimes yield suboptimal solutions. Notwithstanding the marginally suboptimal solutions that are recorded, we do note the marked reduction in solution time.

Remark: The test condition $\sum_j y_{ij} \geq \delta M_i$ implies that the smaller the invalidity test factor, the easier it is to meet the test condition and, in turn, the more likely the algorithm yields the optimal solution. This agrees with the test results, where we recognize that the algorithm converges to the optimal solution only for small values of invalidity test factor δ.

4. Algorithm 3: Results and Analysis

The computational run times are recorded for Algorithm 3 for various values of parameters δ and ϕ, while setting $\gamma = \varphi = 1.1$. Table 8 presents results.

Root Test Factor δ	Tree Test Factor ϕ	Solution Time (s)	Number of Bound Coefficient Updates	Optimality Gap (%)
0.1	0.9	879.5	43	0.0
0.3		891.7	52	0.0
0.5		954.8	49	0.0
0.7		927.5	49	0.0
0.1	0.7	710.5	43	0.0
0.3		834.1	51	0.0
0.5		856.0	49	0.0
0.7		923.0	49	0.0
0.1	0.5	900.4	43	0.0
0.3		1,231.2	50	0.0
0.5		934.7	49	0.0
0.7		933.1	49	0.0
0.1	0.3	892.8	38	0.0
0.3		1,089.5	49	0.0
0.5		928.9	49	0.0
0.7		894.0	49	0.0
Base-case solution time: 1,136 seconds				

Table 8. Solution times for Algorithm 3 on the uncapacitated *k*-median problem

For the various parameters used, Algorithm 3 yields the optimal solution in all instances. This suggests that Algorithm 3, together with this test condition, is insensitive to input parameters. Solution times are also shorter in most cases tested, by up to 37%. This indicates that the bound coefficients have increased sufficiently for validity, but still less than the baseline coefficients.

As the algorithm tests for invalid bound coefficients in the B&B search tree and restarts from the root node whenever updates are made, we conjecture that Algorithm 3, together with this test condition, always yields the optimal solution.

5. Algorithm 3: Results and Analysis with Alternative Test Condition

We proceed to run Algorithm 3 using the alternative test condition for selected values for parameters γ. Again, we set $M_i \leftarrow M_i^I = \sum_j d_j \big/ k$, for all $i \in I$. Table 9 displays the results.

Growth Factor γ	Solution Time (s)	Number of Bound Coefficient Updates	Optimality Gap (%)
1.1	5,405.3	1,446	0.0
1.2	7,208.7	854	0.0
1.3	5,595.1	665	0.0
1.4	5,691.2	665	0.0
1.5	3,879.7	616	0.0
1.6	6,388.3	681	0.0
1.7	5,967.8	665	0.0
1.8	5,877.8	665	0.0
1.9	3,754.0	616	0.0
2.0	7,767.9	692	0.0
Base-case solution time: 1,136 seconds			

Table 9. Solution times for Algorithm 3 (alternative test condition) on the uncapacitated *k*-median problem

We notice that this test condition yields the optimal solution for any choice of growth factor γ. This is consistent with our claim that bound coefficients, at any node, are unsuitable whenever $|S| \neq |J|$, for instances where $c_{ij} \neq c_{i'j}$, for any $i \neq i' \in I$, and any $j \in J$.

Solution times are excessive here, making this invalidity test condition impractical. Even with valid bound coefficients, there may be nodes yielding non-integer solutions that meet the invalidity test condition $|S| \neq |J|$, forcing the bound coefficients to be increased and the problem restarted.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. TEST MODEL II: SHORTEST-PATH INTERDICTION PROBLEM

This chapter describes a SPIP, states the invalidity test condition to be used in Algorithm 3 and solves a problem instance using that algorithm. As will be explained, Algorithms 1 and 2 do not apply here.

A. SHORTEST-PATH INTERDICTION PROBLEM

Given a directed graph $G=(V,E)$, where *V* defines a set of nodes and *E* a set of directed edges connecting the nodes, the SPIP involves a network user, or *defender*, and an *attacker*. The defender seeks to traverse a path of minimum length between two specified nodes, *s* and *t*. The attacker seeks to maximize the shortest-path length of the defender, and does this by interdicting edges in the network and increasing the effective lengths of the interdicted edges, or by making those edges impassable. We assume that interdiction of edge (i, j) does not imply interdiction of edge (j,i), if that edge exists.

Fulkerson and Harding (1977) study a version of the SPIP, where they allow the penalty on a particular edge to increase as a linear function of resources allocated to the interdiction of that edge. Israeli and Wood (2002) study the SPIP with binary interdiction effort, i.e., with fixed penalties; they propose two algorithms to solve it. (See also Bayrak & Bailey, 2008.)

We solve the SPIP with binary interdiction effort, formulated as, follows:

Indices

V set of nodes

E set of edges

$i \in V$ nodes in *G*, where *s* is the source node, *t* is the sink node

$k = (i, j) \in E$ edges in *G*, where *i* is the origin and *j* is the destination node of edge *k*

$k \in FS(i)$ $(k \in RS(i))$ edges directed out of (into) node *i*

Parameters

c_k cost to traverse edge *k*

M_k added delay if edge *k* is interdicted

Variables

x_k 1 if edge *k* is interdicted by the attacker, 0 otherwise

y_k 1 if edge *k* is part of the post-interdiction shortest path, 0 otherwise

Formulation

$$z^* = \max_{\mathbf{x} \in X} \min_{\mathbf{y}} \sum_k (c_k + x_k M_k) y_k \tag{4.A.1}$$

$$\text{s.t.} \sum_{k \in FS(i)} y_k - \sum_{k \in RS(i)} y_k = \begin{cases} 1, & i = s \\ 0, & \forall i \in V \setminus \{s,t\} \\ -1, & i = t \end{cases} \tag{4.A.2}$$

$$\mathbf{y} \geq 0 \tag{4.A.3}$$

$$\mathbf{x} \in \{0,1\}^{|E|} \tag{4.A.4}$$

Here, X defines the set of resources available to conduct interdictions.

The bound coefficient of interest here is the penalty cost M_k. Setting $M_k = |V| \max_{k'} \{c_{k'}\}$ for all $k \in E$ ensures that interdiction of an edge makes that edge effectively impassable: This is the problem variant we wish to test.

The objective function (4.A.1) seeks to maximize the minimum cost for the defender to traverse the graph. Flow-balance constraints are modeled in (4.A.2) to route one unit of flow from *s* to *t*. While y_k is required to be binary, the unimodular property of the shortest-path problem allows this integer requirement to be relaxed and replaced with a non-negativity requirement.

One approach to solve the SPIP is to take the dual of the inner minimization problem (Fulkerson & Harding 1977, Israeli & Wood 2002). The following MIP results:

$$z^* = \max_{\mathbf{x}\in X,\pi} \ \pi_t - \pi_s \tag{4.A.5}$$

$$\text{s.t.} \ \ \pi_j - \pi_i \le c_k + M_k x_k, \ \ \forall k = (i, j) \in E \tag{4.A.6}$$

$$\pi_s = 0 \tag{4.A.7}$$

$$\mathbf{x} \in \{0,1\}^{|E|} \tag{4.A.8}$$

Here, π_i denotes the unrestricted dual variable associated with equation (4.A.2). However, we may assume $\pi_s = 0$ as the inner minimization problem has at least one redundant flow-balance constraint. And, once we fix $\pi_s = 0$, we can also assume $\pi_i \ge 0$ for all $i \in V$. A standard B&B algorithm will solve the SPIP.

B. TEST CONDITION FOR SHORTEST-PATH INTERDICTION PROBLEM

Given our interpretation of interdiction, a value of M_k is invalid if an edge is interdicted and, at the same time, flow is being sent on that edge. The interdiction plan is obtained from the solution of the MIP shown in (4.A.5) – (4.A.8) (i.e., the values of the variables x_k), and flows in the network are obtained from the dual variables of equation (4.A.6) (which is essentially the flow variable y_k in the primal subproblem). As before, I_n^v is defined to be the index set of coefficients M_i that meet the invalidity test condition at node *n*. For Algorithm 3, for the SPIP, the invalidity test condition is

$$k = (i, j) \in I_n^v \text{ if } x_k = 1 \text{ and } y_k > 0. \tag{4.B.1}$$

Increase factors γ and φ are defined so that $M_k \leftarrow \gamma M_k$ for all $k \in I_0^v$, and $M_k \leftarrow \varphi M_k$, for all $k \in I_n^v$ with $n \ge 1$. Furthermore, if either $\gamma M_k \ge |V| \max_{k'}\{c_{k'}\}$ or $\varphi M_k \ge |V| \max_{k'}\{c_{k'}\}$, we let $M_k \leftarrow |V| \max_{k'}\{c_{k'}\}$.

We test the SPIP only with Algorithm 3. At the time of writing, no inefficiency test condition has been developed; hence Algorithm 1 is not implemented. Also, it is unlikely the root node will yield integer interdiction solutions; hence Algorithm 2 is not implemented as the invalidity test condition will not be met at the root node.

C. COMPUTATIONAL RESULTS

We test Algorithm 3 on an instance of SPIP with $|V| = 102$ and $|E| = 416$. While testing is limited to one instance only, it should serve as a proof-of-concept if good results are obtained, i.e., further investigation will be warranted. For the purpose of this test model, costs to traverse edges are randomly generated according to a uniform distribution between 1 and 10; actual values used are listed in Appendix B.

1. Solution by Standard Branch and Bound Algorithm

The test problem is first run using the standard B&B algorithm, once with baseline (weak) coefficients, and once with tighter, empirically valid, bound coefficients. The results are shown in Table 12.

Case	Bound Coefficient, M_k	Solution Time (s)	Optimality Gap (%)
Base-case: Weak Bound Coefficient	$\|V\| \max_{k'}\{c_{k'}\}$	186.1	0.0
Tight Bound Coefficient	$0.01\|V\| \max_{k'}\{c_{k'}\}$	0.4	0.0

Table 10. Solution times for standard B&B on SPIP

We again see that having weak bound coefficients can be detrimental to solution times.

2. Algorithm 3: Results and Analysis

We proceed to run Algorithm 3 for select values of γ and φ, where we choose $\gamma = \varphi$ as both are increase factors, albeit at different stages of the algorithm. We also set $M_k^I = 0.001|V|\max_{k'}\{c_{k'}\}$, for all $k \in E$. Table 13 presents results.

Growth Factors γ, φ	Solution Time (s)	Number of Penalty Updates	Optimality Gap (%)
1.1	1.1	38	0.0
1.2	0.6	20	3.0
1.3	0.4	14	3.0
1.4	0.5	9	3.0
1.5	0.3	9	3.0
1.6	0.5	13	3.0
1.7	0.3	11	3.0
1.8	0.4	11	0.0
1.9	0.5	10	3.0
2.0	0.4	8	3.0
2.1	0.5	9	3.0
2.2	0.4	8	3.0
2.3	0.5	9	0.0
2.4	0.4	8	3.0
2.5	0.5	10	0.0
2.6	0.5	8	3.0
2.7	0.4	8	0.0
2.8	0.6	12	3.0
2.9	0.6	11	0.0
3.0	0.3	6	3.0
Base-case run time: 186.1 seconds			

Table 11. Solution times for Algorithm 3 on SPIP

Algorithm 3 with the described test condition does not always yield an optimal solution. However, for the cases tested, the optimality gap is at most 3%, and solution

times are reduced by at least 99.4%. While optimality is not guaranteed with this test condition, the marked reduction in solution times warrants further investigation.

3. Additional Remarks

Xpress-IVE, with default settings enabled, solves the SPIP defined with baseline coefficients in 1.3 seconds. This indicates that the efficiency provided by presolving techniques, cutting-plane strategies and heuristics cannot be ignored. We have not carried out the analogous test for our version of the uncapacitated *k*-median problem; however: some solvers actually transform the weak formulation into a stronger one before solving, and since we do not know how Xpress-IVE would handle our formulation, test results would not be meaningful.

V. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

The continuous relaxation of a MIP containing standard big-*M* constructs can be weak because large values of *M* ("bound coefficients") are required to ensure the model's validity. This, in turn, may lead to slow convergence when solving the model using a standard B&B algorithm. Standard big-*M* constructs are used in MIPs such as (a) the weak formulations of the uncapacitated *k*-median problem and UFLP, (b) the production-planning problem and (c) the SPIP. This thesis develops three algorithms to adaptively adjust bound coefficients within the B&B tree. The goal is to achieve optimal solutions faster.

Algorithm 1 starts with initially weak bound coefficients defined by standard techniques (baseline coefficients), and reduces them within the B&B search tree. Algorithms 2 and 3 start with initially small and possibly invalid coefficients and increase them as their respective B&B algorithms proceed. Algorithm 2 detects potentially invalid coefficients at the root node of the B&B search tree only, while Algorithm 3 detects such coefficients throughout the B&B tree. When coefficients are increased for Algorithms 2 and 3, the problem is re-solved from the root node. Central to all three algorithms is the need for *test conditions* that signify possible invalidity or inefficiency of bound coefficients. Algorithms 1, 2 and 3 are tested on an instance of the uncapacitated *k*-median problem, and Algorithm 3 is also tested on an instance of the SPIP.

The uncapacitated *k*-median problem seeks to place production facilities so that customer demands for a single product can be met at minimum cost, while placing a limit on the total number of facilities to be opened. For testing purposes, we use a weak formulation in which each bound coefficient limits the total amount of product that a facility can ship to customers. In the instances where Algorithms 1 and 2 yield an optimal solution, we observe a reduction in solution time of up to 75% and 62%, respectively. Importantly, when Algorithms 1 and 2 yield marginally suboptimal solutions, substantial solution-time improvements are recorded. Algorithm 3 yields some

promising results: All cases tested converge to an optimal solution, with up to a 37% decrease in solution time. An alternative test condition for Algorithm 3 also yields an optimal solution in all cases, but with excessive solution times.

The SPIP seeks to use limited resources to interdict (destroy) arcs in a network in order to maximize the length (cost) of the shortest (least-cost) path between two nodes. In the tested instances, bound coefficients refer to the penalties required to model edges as effectively impassable when interdicted. Algorithms 1 and 2 are not suited for the SPIP, so only Algorithm 3 is tested. Results show an optimality gap of at most 3%, with solution times reduced by at least 99.4%.

Notwithstanding the promising results from the test models, further research must be conducted before practical implementations can be adopted.

B. RECOMMENDATIONS FOR FUTURE RESEARCH

This research area is open to further developments of the algorithms. Algorithm 3 can be improved further such that re-solving a problem is not required every time bound coefficients are updated. This will involve reexamining fathomed B&B nodes whenever bound coefficients are updated, in case a node was fathomed incorrectly.

Another potential area for future research would combine the ideas behind Algorithms 1 and 3 to create a single algorithm that is able to detect and update both invalid and weak bound coefficients. This would allow the user to initialize bound coefficients to nearly arbitrary positive values.

Other areas of focus are formulation of algorithms that are provably correct and yield substantial reductions in run times. It is possible this may not include the ideas used in the adaptive B&B algorithms.

The algorithms proposed in this thesis are tested on instances of the uncapacitated *k*-median problem and SPIP with randomly generated data. Real-world problems, such as production lot-sizing problems should be identified and used to test the algorithms.

LIST OF REFERENCES

Balinski, M. (1966). On finding integer solutions to linear programs. *Proceedings of the IBM Scientific Computing Symposium on Combinatorial Problems*, (225–248), New York: IBM.

Bayrak, H., & Bailey, M. (2008). Shortest path network interdiction with asymmetric information. *Networks*, *52*, 133–140.

Brearley, A., Mitra, G., & Williams, H. (1975). Analysis of mathematical programming problems prior to applying the simplex algorithm. *Mathematical Programming*, *8*(1), 54–83.

Brown, G., Carlyle, M., Salmerón, J., & Wood, K. (2006). Defending critical infrastructure. *Interfaces*, *36*(6), 530–544.

Camm , J., Raturi, A., & Tsubakitani, S. (1990). Cutting big *M* down to size. *Interfaces*, *20*(5), 61–66.

Chinneck, J. (2002). Discovering the characteristics of mathematical programs via sampling. *Optimization Methods and Software*, *17*, 319–352.

Crowder, H., Johnson, E., & Padberg, M. (1983). Solving large-scale zero-one linear programming problems. *Operations Research*, *31*, 803–834.

Efroymson, M., & Ray, T. (1966). A branch and bound algorithm for plant location. *Operation Research*, *14*(3), 361–368.

Eppen, G., & Martin, R. (1987). Solving multi-item capacitated lot-sizing problems using variable redefinition. *Operations Research*, *6*, 832–848.

Fulkerson, D., & Harding, G. (1977). Maximizing the minimun source-sink path subject to a budget constraint. *Mathematical Programming*, *13*, 116–118.

Israeli, E., & Wood, R. (2002). Shortest-path network interdiction. *Networks*, *40*, 97–111.

Johnson, E., Kostreva, M., & Suhl, V. (1985). Solving 0-1 integer programming problems arising from large-scale planning models. *Operations Research*, *33*(4), 803–819.

Khumawala, B. (1972). An efficient branch and bound algorithm for the warehouse location problem. *Management Science*, *18*(12), B718–B731.

Lawler, E., & Wood, D. (1966). Branch-and-bound methods: A survey. *Operations Research*, *14*(4), 699–719.

Lin, J. H., & Vitter, J. S. (1992). ε-approximations with minimum packing constraint violation. *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, (771–782), New York: ACM.

Maes, J., & Van Wassenhove, L. (1988). Multi-item single-level capacitated dynamic lot-sizing heuristics: A general review. *The Journal of the Operational Research Society*, *39*, 991–1004.

Markland, R., & Sweigart, J. (1987). *Quantitative methods: Applications to managerial decision making.* New York: John Wiley and Sons.

Rardin, R. (1998). *Optimization in operations research.* Upper Saddle River, NJ: Prentice Hall, Inc.

Reese, J. (2005). *Methods for solving the p-median problem: An annotated bibliography.* San Antonio, TX: Trinity University.

Van Roy, T., & Wolsey, L. (1987). Solving mixed integer programming problems using automatic reformulation. *Operations Research*, *35*, 45–57.

Wood, R. (1993). Deterministic network interdiction. *Mathematical and Computer Modelling*, *17*(2), 1–18.

APPENDIX A. DISTRIBUTION OF DATA FOR UNCAPACITATED *k*-MEDIAN TEST MODEL

The uncapacitated *k*-median problem tested comprises n = 50 locations, from which k = 5 must be chosen to fulfill the demands of m = 100 customers. The data used are randomly generated according a uniform distribution using random seed equal to 2 in Xpress-IVE. The details are as follows:

- $(xFac_i, yFac_i)$ denotes the (x,y) coordinates for facility i, where $xFac_i \sim Uniform(0,3000)$ and $yFac_i \sim Uniform(0,1000)$.
- $(xCust_j, yCust_j)$ denotes the (x,y) coordinates for customer j, where $xCust_j \sim Uniform(0,3000)$ and $yCust_j \sim Uniform(0,1000)$.
- c_{ij} denotes per unit shipping cost from facility i to customer j, where $c_{i,j} = \sqrt{\left(xFac_i - xCust_j\right)^2 + \left(yFac_i - yCust_j\right)^2}$
- d_j denotes units of demand at customer j, where $d_j \sim Uniform(100,200)$
- p_j denotes penalty per unit of unmet demand at customer j, where $p_j = 1.1 \max_{i,j}\{c_{ij}\}$

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. DATA SET FOR SHORTEST-PATH INTERDICTION PROBLEM

The SPIP test model uses an asymmetric grid network $G = (V, E)$ with $|V| = 102$ and $|E| = 416$. Table 14 presents all of the data for the test instance, except the facts that $s = 1$, $t = 2$.

Arc	Cost	Arc	Cost	Arc	Cost	Arc	Cost
(1,3)	3	(27,37)	10	(49,58)	3	(72,82)	1
(3,13)	6	(27,38)	4	(50,49)	9	(72,81)	3
(3,14)	9	(27,36)	10	(50,51)	3	(73,74)	10
(1,4)	9	(28,27)	10	(50,60)	2	(73,83)	2
(4,14)	10	(28,29)	7	(50,61)	6	(73,84)	1
(4,15)	7	(28,38)	2	(50,59)	6	(74,73)	8
(4,13)	2	(28,39)	4	(51,50)	10	(74,75)	7
(1,5)	6	(28,37)	10	(51,52)	6	(74,84)	4
(5,15)	2	(29,28)	3	(51,61)	9	(74,85)	10
(5,16)	5	(29,30)	4	(51,62)	1	(74,83)	9
(5,14)	9	(29,39)	2	(51,60)	10	(75,74)	6
(1,6)	7	(29,40)	4	(52,51)	5	(75,76)	5
(6,16)	8	(29,38)	1	(52,62)	2	(75,85)	6
(6,17)	9	(30,29)	7	(52,61)	5	(75,86)	7
(6,15)	3	(30,31)	8	(53,54)	8	(75,84)	7
(1,7)	5	(30,40)	10	(53,63)	3	(76,75)	6
(7,17)	3	(30,41)	7	(53,64)	3	(76,77)	9
(7,18)	5	(30,39)	5	(54,53)	6	(76,86)	8
(7,16)	10	(31,30)	2	(54,55)	1	(76,87)	7
(1,8)	10	(31,32)	5	(54,64)	4	(76,85)	10
(8,18)	9	(31,41)	2	(54,65)	7	(77,76)	5
(8,19)	1	(31,42)	6	(54,63)	1	(77,78)	5
(8,17)	8	(31,40)	5	(55,54)	3	(77,87)	9
(1,9)	6	(32,31)	1	(55,56)	3	(77,88)	4
(9,19)	2	(32,42)	4	(55,65)	4	(77,86)	1
(9,20)	6	(32,41)	6	(55,66)	4	(78,77)	1
(9,18)	9	(33,34)	7	(55,64)	6	(78,79)	4
(1,10)	6	(33,43)	9	(56,55)	4	(78,88)	3
(10,20)	9	(33,44)	7	(56,57)	9	(78,89)	8

Arc	Cost	Arc	Cost	Arc	Cost	Arc	Cost
(10,21)	3	(34,33)	10	(56,66)	3	(78,87)	7
(10,19)	8	(34,35)	7	(56,67)	9	(79,78)	4
(1,11)	2	(34,44)	3	(56,65)	7	(79,80)	5
(11,21)	8	(34,45)	9	(57,56)	10	(79,89)	7
(11,22)	6	(34,43)	6	(57,58)	1	(79,90)	1
(11,20)	2	(35,34)	2	(57,67)	6	(79,88)	3
(1,12)	9	(35,36)	9	(57,68)	7	(80,79)	3
(12,22)	10	(35,45)	8	(57,66)	7	(80,81)	9
(12,21)	1	(35,46)	2	(58,57)	4	(80,90)	6
(13,14)	10	(35,44)	5	(58,59)	3	(80,91)	9
(13,23)	7	(36,35)	9	(58,68)	6	(80,89)	1
(13,24)	1	(36,37)	8	(58,69)	10	(81,80)	2
(14,13)	4	(36,46)	7	(58,67)	8	(81,82)	10
(14,15)	5	(36,47)	4	(59,58)	1	(81,91)	8
(14,24)	6	(36,45)	1	(59,60)	8	(81,92)	5
(14,25)	6	(37,36)	1	(59,69)	10	(81,90)	1
(14,23)	10	(37,38)	3	(59,70)	2	(82,81)	1
(15,14)	5	(37,47)	10	(59,68)	9	(82,92)	5
(15,16)	3	(37,48)	9	(60,59)	10	(82,91)	10
(15,25)	10	(37,46)	5	(60,61)	4	(83,84)	7
(15,26)	6	(38,37)	5	(60,70)	2	(83,93)	2
(15,24)	7	(38,39)	8	(60,71)	1	(83,94)	3
(16,15)	8	(38,48)	9	(60,69)	10	(84,83)	10
(16,17)	4	(38,49)	1	(61,60)	8	(84,85)	7
(16,26)	2	(38,47)	7	(61,62)	3	(84,94)	3
(16,27)	3	(39,38)	3	(61,71)	3	(84,95)	1
(16,25)	6	(39,40)	9	(61,72)	3	(84,93)	6
(17,16)	7	(39,49)	6	(61,70)	10	(85,84)	8
(17,18)	8	(39,50)	3	(62,61)	5	(85,86)	8
(17,27)	5	(39,48)	2	(62,72)	6	(85,95)	3
(17,28)	7	(40,39)	9	(62,71)	8	(85,96)	4
(17,26)	8	(40,41)	5	(63,64)	9	(85,94)	1
(18,17)	7	(40,50)	5	(63,73)	5	(86,85)	3
(18,19)	1	(40,51)	7	(63,74)	1	(86,87)	7
(18,28)	9	(40,49)	8	(64,63)	5	(86,96)	6
(18,29)	3	(41,40)	6	(64,65)	10	(86,97)	7
(18,27)	9	(41,42)	10	(64,74)	3	(86,95)	4
(19,18)	2	(41,51)	2	(64,75)	9	(87,86)	3

Arc	Cost	Arc	Cost	Arc	Cost	Arc	Cost
(19,20)	6	(41,52)	5	(64,73)	5	(87,88)	8
(19,29)	7	(41,50)	7	(65,64)	4	(87,97)	7
(19,30)	3	(42,41)	2	(65,66)	3	(87,98)	3
(19,28)	6	(42,52)	4	(65,75)	10	(87,96)	3
(20,19)	7	(42,51)	4	(65,76)	10	(88,87)	6
(20,21)	10	(43,44)	2	(65,74)	2	(88,89)	9
(20,30)	4	(43,53)	8	(66,65)	9	(88,98)	3
(20,31)	1	(43,54)	7	(66,67)	10	(88,99)	10
(20,29)	5	(44,43)	6	(66,76)	9	(88,97)	8
(21,20)	5	(44,45)	4	(66,77)	4	(89,88)	9
(21,22)	2	(44,54)	9	(66,75)	7	(89,90)	2
(21,31)	9	(44,55)	6	(67,66)	5	(89,99)	1
(21,32)	3	(44,53)	6	(67,68)	9	(89,100)	3
(21,30)	10	(45,44)	10	(67,77)	7	(89,98)	8
(22,21)	1	(45,46)	7	(67,78)	10	(90,89)	10
(22,32)	7	(45,55)	3	(67,76)	2	(90,91)	8
(22,31)	8	(45,56)	4	(68,67)	2	(90,100)	7
(23,24)	4	(45,54)	2	(68,69)	8	(90,101)	3
(23,33)	10	(46,45)	3	(68,78)	2	(90,99)	5
(23,34)	6	(46,47)	7	(68,79)	4	(91,90)	1
(24,23)	5	(46,56)	1	(68,77)	6	(91,92)	2
(24,25)	7	(46,57)	6	(69,68)	6	(91,101)	10
(24,34)	6	(46,55)	9	(69,70)	10	(91,102)	7
(24,35)	2	(47,46)	9	(69,79)	5	(91,100)	3
(24,33)	6	(47,48)	9	(69,80)	2	(92,91)	10
(25,24)	7	(47,57)	9	(69,78)	8	(92,102)	2
(25,26)	10	(47,58)	5	(70,69)	9	(92,101)	2
(25,35)	8	(47,56)	8	(70,71)	4	(93,2)	2
(25,36)	3	(48,47)	10	(70,80)	3	(94,2)	3
(25,34)	2	(48,49)	8	(70,81)	1	(95,2)	5
(26,25)	8	(48,58)	5	(70,79)	6	(96,2)	4
(26,27)	2	(48,59)	3	(71,70)	4	(97,2)	10
(26,36)	3	(48,57)	5	(71,72)	2	(98,2)	9
(26,37)	7	(49,48)	1	(71,81)	5	(99,2)	8
(26,35)	10	(49,50)	2	(71,82)	4	(100,2)	10
(27,26)	4	(49,59)	5	(71,80)	2	(101,2)	4
(27,28)	7	(49,60)	1	(72,71)	5	(102,2)	8

Table 12. Data for SPIP test problem

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Dr. Robert F. Dell
   Naval Postgraduate School
   Monterey, California

4. Distinguished Professor R. Kevin Wood
   Naval Postgraduate School
   Monterey, California

5. Associate Professor Javier Salmerón
   Naval Postgraduate School
   Monterey, California

6. Assistant Professor Nedialko Dimitrov
   Naval Postgraduate School
   Monterey, California

7. Professor Yeo Tat Soon
   Director, Temasek Defence Systems Institute
   National University of Singapore, Singapore

8. Ms. Tan Lai Poh
   Senior Manager, Temasek Defence Systems Institute
   National University of Singapore, Singapore

9. COL Lim Soon Chia
   Dy CRTO, DRTech
   Singapore

10. Russell Gan
    Singapore Army
    Singapore